**Micro-services structure**

**Link Github:**

Gateway: <https://github.com/hafbn/Microservices-gateway.git>

App: <https://github.com/hafbn/microservicesApp.git>

Generate a micro-services application using JHipster:

Java version: jdk-11.0.16.1

**General structure**: In a microservices application, services are independent, self-contained components that are responsible for a specific business capability. Each service is typically implemented as a separate process, running in its own container, and communicates with other services through APIs.

* Service registry: a service registry used to manage and discover the various microservices with theirs URLs that make up the application.
* Microservices application: the app, implemented with Spring Boot
* API Gateway: entry point, implemented with Spring Cloud Gateway
* Database: relational database, generated automatically based on model created when define entities.

First, generate microservices registry:

* Install JHipster
* As JHipster Registry is a standard JHipster, we clone it on GitHub:

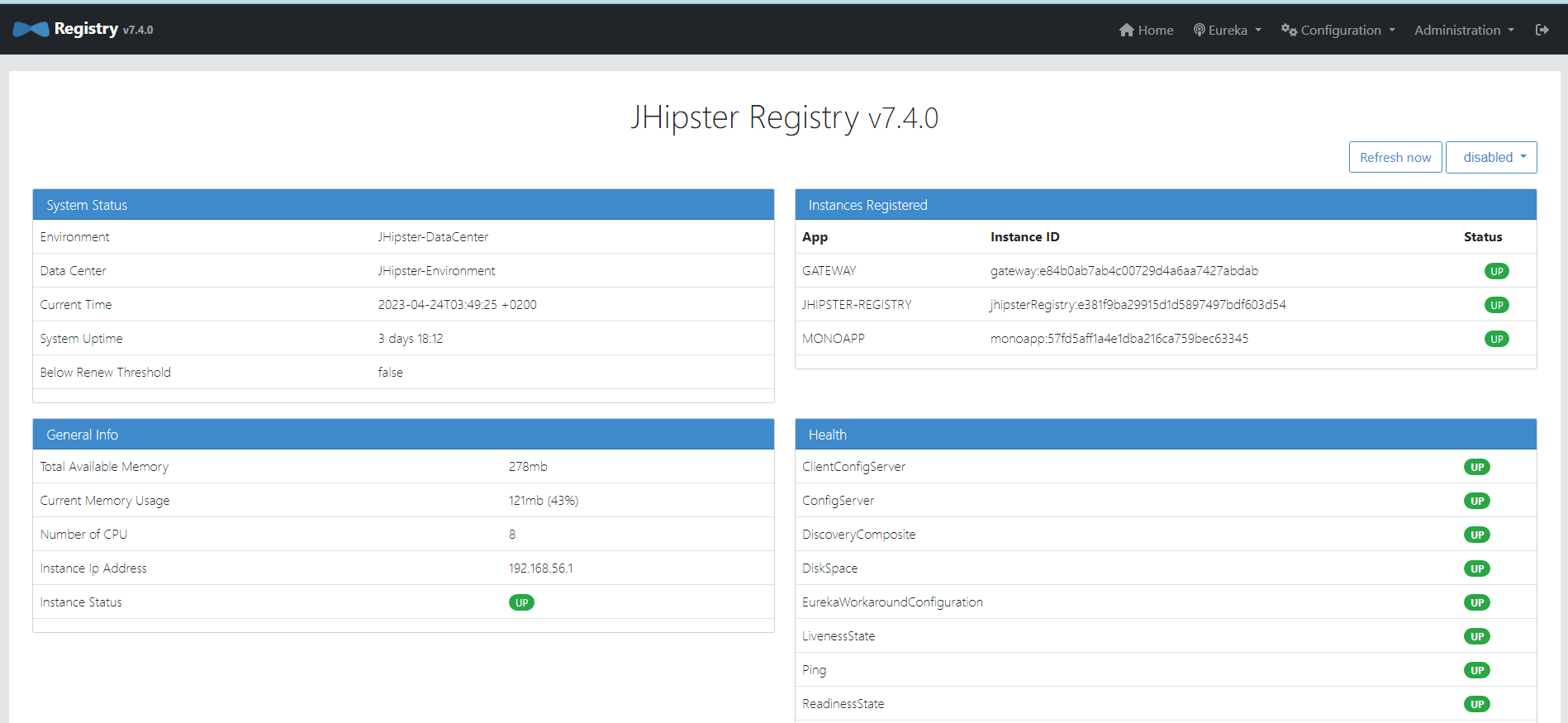
***git clone https://github.com/jhipster/jhipster-registry***

* Then jump to ***jhipster-registry*** folder and run the app:

***cd jhipster-registry***

***mvnw***

* Registry is up on <http://localhost:8761/>

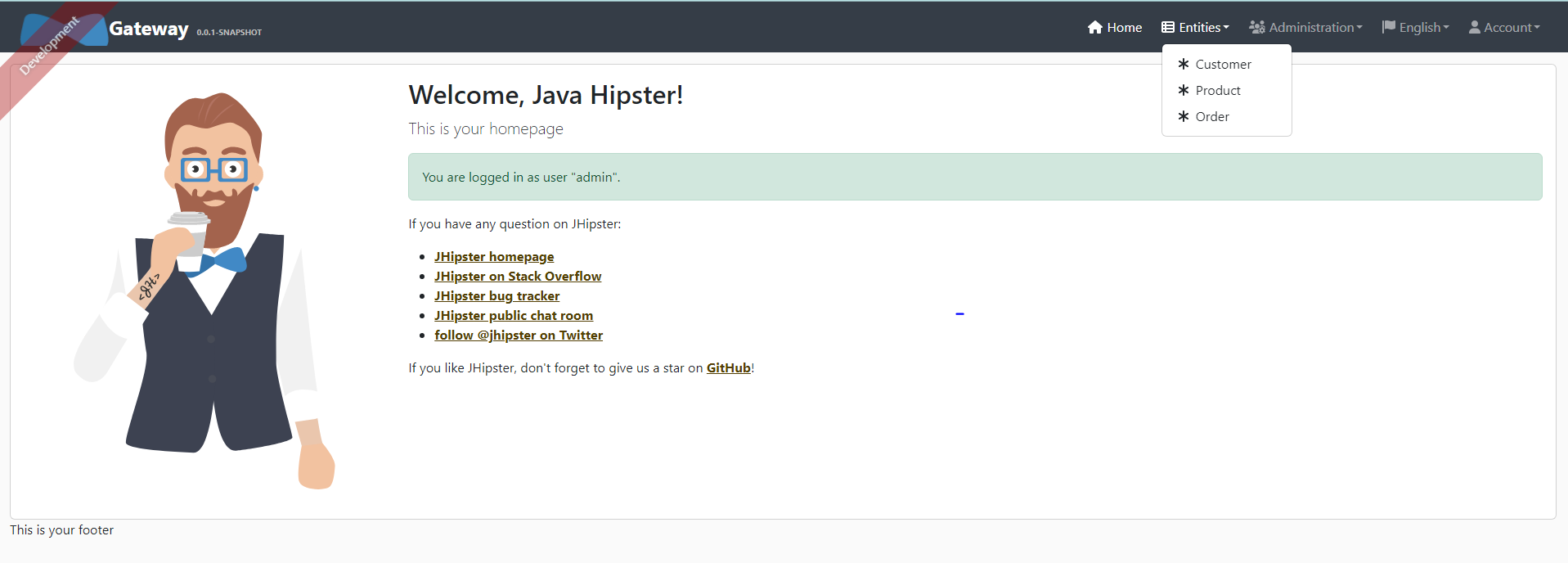


Next, generate a Microservices app:

* Create a folder for the app: ***/blog***
* Jump to the folder and run ***jhipster*** to generate a Microservice app with the following options:
* Port: 8081
* Database: SQL (MySQL)
* Maven to build backend
* Add ***Customer, Product & Order*** entities using jhipster entity …
* Run app using ***mvnw***, app is up on registry

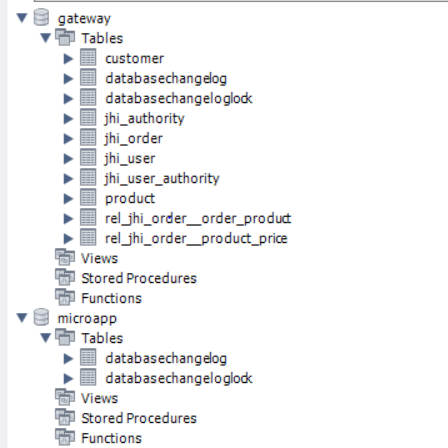
Then, generate Microservices Gateway:

* Create a folder for gateway app: ***/gateway***
* Jump to the folder and run ***jhipster*** to create gateway app with the following options:
* Port: 8080
* Service discovery & config: Jhipster registry
* Database: SQL (MySQL)
* Maven to build backend
* ReactJS for frontend
* Run ***mvnw***, gateway is up on <http://localhost:8080/> with all the entites created, login with admin



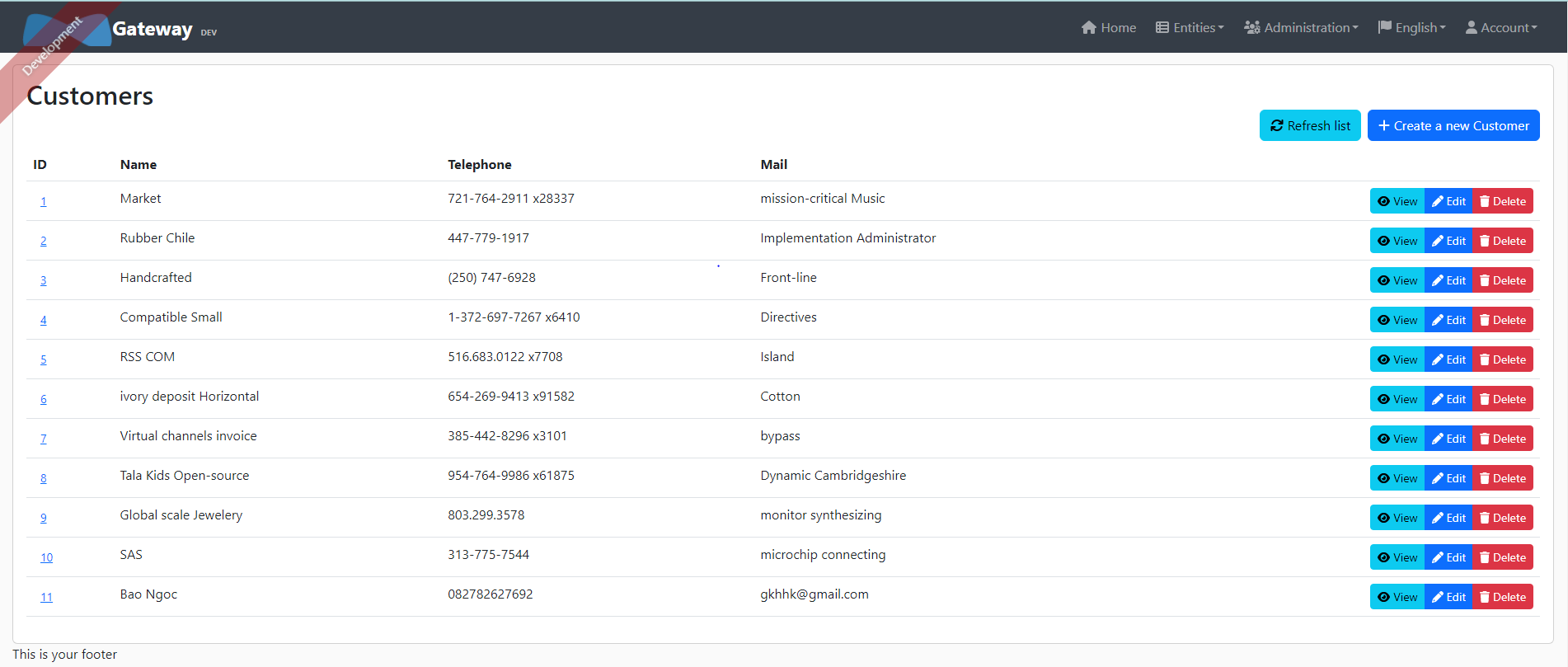
To run frontend app separately, jump to ***src/main/webpack*** and run ***npm start*** for realtime reload. Front app is up on <http://localhost:9000/>

Database is created and stocked in 2 schemas: ***app*** & ***gateway***:



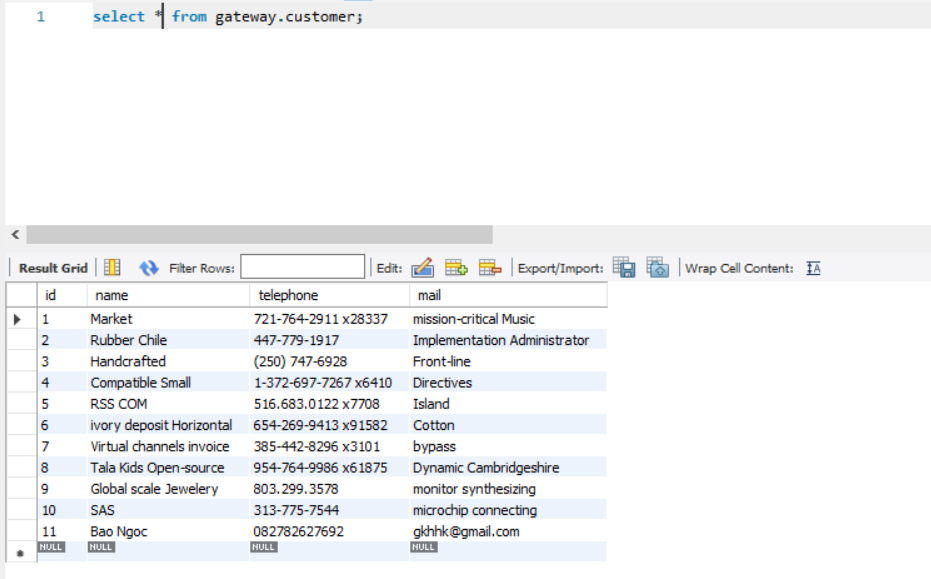
**Entities**

* Customer

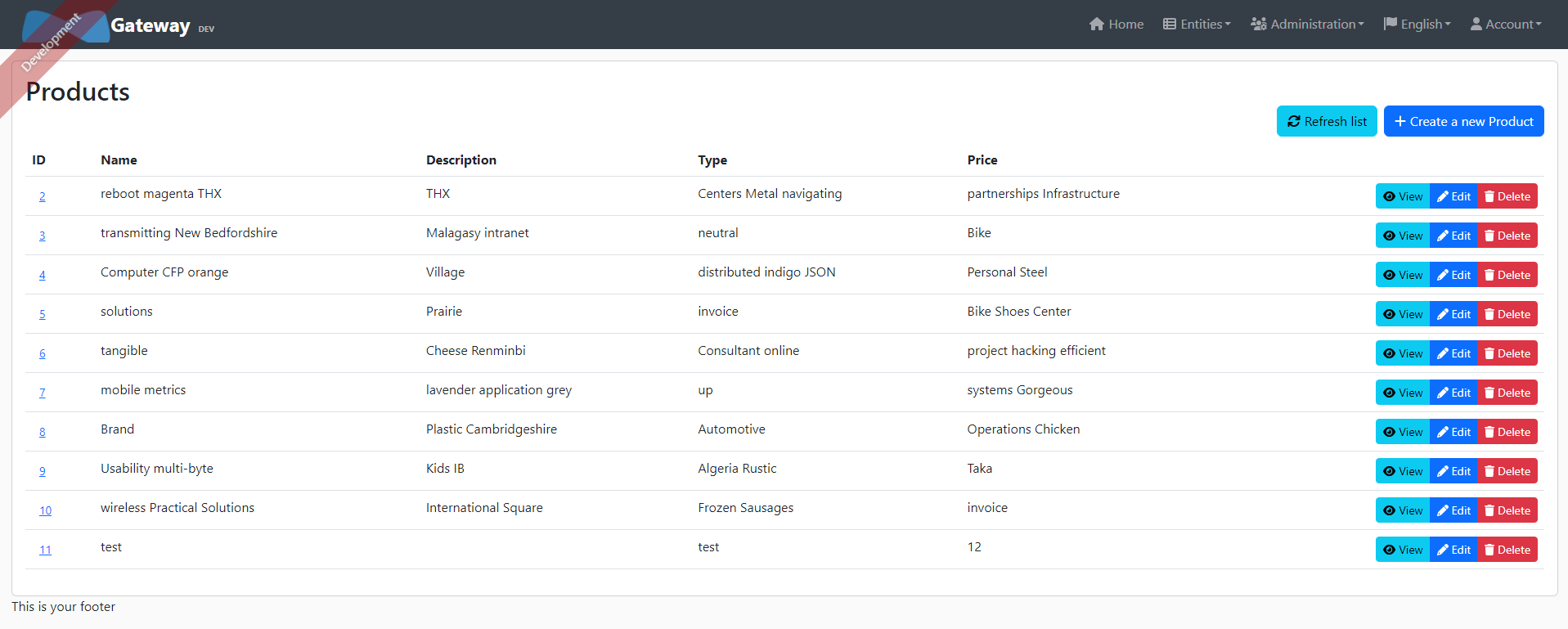


Customer is created with 4 fields: ID, name, telephone & mail. All fields are required, ID & name are unique. We can create, view, edit and delete a customer.

Every change on customer list will be noticed in database, example in MySQL Workbench:



* Product



Product is created with 5 fields: ID, name, description, type & price. ID, price & type are required, only ID is unique.

* Order (not done)

**APIs**

When navigate to /swagger-ui/index.html, we see list of APIs to connect all the services:

